Fast calculation of Greeks by Monte Carlo using adjoint methods

Mike Giles and Paul Glasserman
giles@comlab.ox.ac.uk

http://web.comlab.ox.ac.uk/mike.giles/finance.html

Oxford University Computing Laboratory

Columbia Business School

Outline

- Monte Carlo sensitivity analysis
- Adjoint evaluation of pathwise sensitivities
  - general concept
  - LMM implementation
  - algorithmic/automatic differentiation
- Conclusions and future work

Generic Monte Carlo

Consider the expected value of a discounted payoff

$$E[g(X(T)],$$

given that $X(t)$ satisfies a stochastic differential equation

$$\mathrm{d}X(t) = a(X(t))\,\mathrm{d}t + b(X(t))\,\mathrm{d}W(t),$$

where X is m-dimensional and W is a d-dimensional Brownian motion.

For hedging and risk analysis, we are interested in the sensitivity to multiple parameters. The question is how to calculate these accurately and efficiently?

LIBOR Market Model

As an example, consider the LIBOR market model of BGM, with $m+1$ bond maturities T_i, with spacings $T_{i+1} - T_i = \delta_i$.

The forward rate for the interval $[T_i, T_{i+1})$ satisfies

$$\frac{\mathrm{d}L_i(t)}{L_i(t)} = \mu_i(L(t))\,\mathrm{d}t + \sigma_i^\top\,\mathrm{d}W(t), \quad 0 \le t \le T_i,$$

where
$$\mu_i(L(t)) = \sum_{j=\eta(t)}^{i} \frac{\sigma_i^\top \sigma_j\,\delta_j L_j(t)}{1+\delta_j L_j(t)},$$

and $\eta(t)$ is the index of the next maturity date.

For simplicity, we keep $L_i(t)$ constant for $t > T_i$, and take the volatilities to be a function of the time to maturity,

$$\sigma_i(t) = \sigma_{i-\eta(t)+1}(0).$$

Finite difference sensitivities

If $V(\theta) = E[g(X(T))]$ for a particular value of an input parameter θ, then

$$\frac{\partial V}{\partial \theta} \approx \frac{V(\theta+\Delta\theta) - V(\theta)}{\Delta\theta}$$

Pros:

- simple to implement

Cons:

- expensive (1 extra calculation for each sensitivity)
- bias error if $\Delta\theta$ too large
- large variance if $g(X(T))$ discontinuous and $\Delta\theta$ small

Pathwise sensitivities

Under certain conditions,

$$\frac{\partial}{\partial\theta}\, E[g(X(T))] = E\left[\frac{\partial g(X(T))}{\partial\theta}\right] = E\left[\frac{\partial g}{\partial X}\,\frac{\partial X(T)}{\partial\theta}\right].$$

with $\dfrac{\partial X(T)}{\partial\theta}$ computed by differentiating the path evolution.

Pros:

- less expensive (1 cheap calculation for each sensitivity)
- no bias

Cons:

- more difficult to implement

Likelihood ratio method

Defining $p(X)$ to the terminal probability density, then

$$V = E[g(X(T))] = \int g(X)\, p(X)\, \mathrm{d}X,$$

and hence

$$\frac{\partial V}{\partial \theta} = \int g \frac{\partial p}{\partial \theta}\, \mathrm{d}X = \int g \frac{\partial (\log p)}{\partial \theta}\, p\, \mathrm{d}X = E\left[g \frac{\partial (\log p)}{\partial \theta} \right].$$

Pros:

- can handle discontinuous payoffs

Cons:

- needs terminal probability density – if not known, additional approximations may be required.

LMM results

Above has been by way of an introduction – now present numerical results to motivate the remainder of the talk.

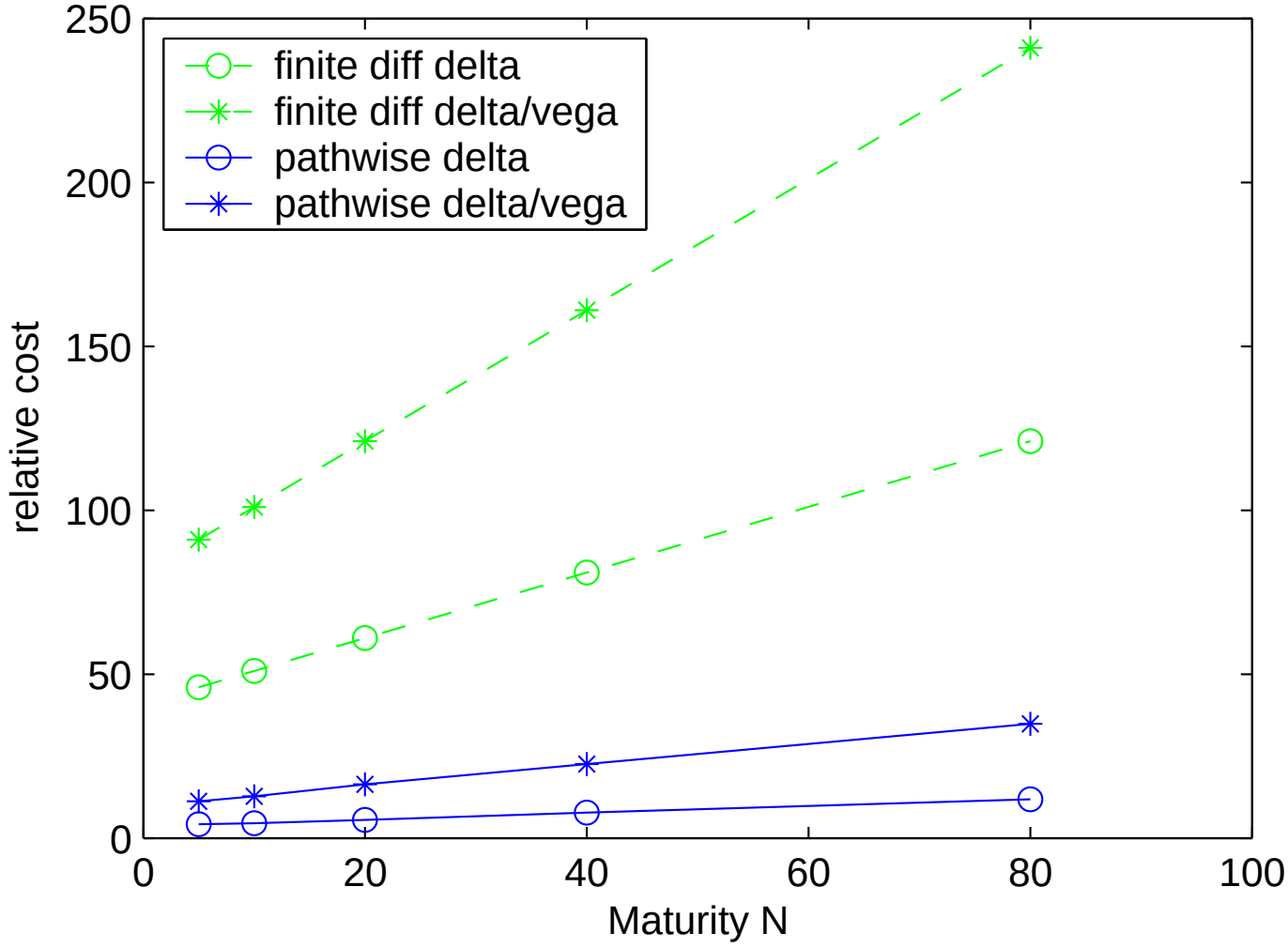
LMM portfolio has 15 swaptions all expiring at the same time, N periods in the future, involving payments/rates over an additional 40 periods in the future.

Interested in computing Deltas, sensitivity to initial $N+40$ forward rates, and Vegas, sensitivity to initial $N+40$ volatilities.

Focus is on the cost of calculating the portfolio value and the sensitivities, relative to just the value.

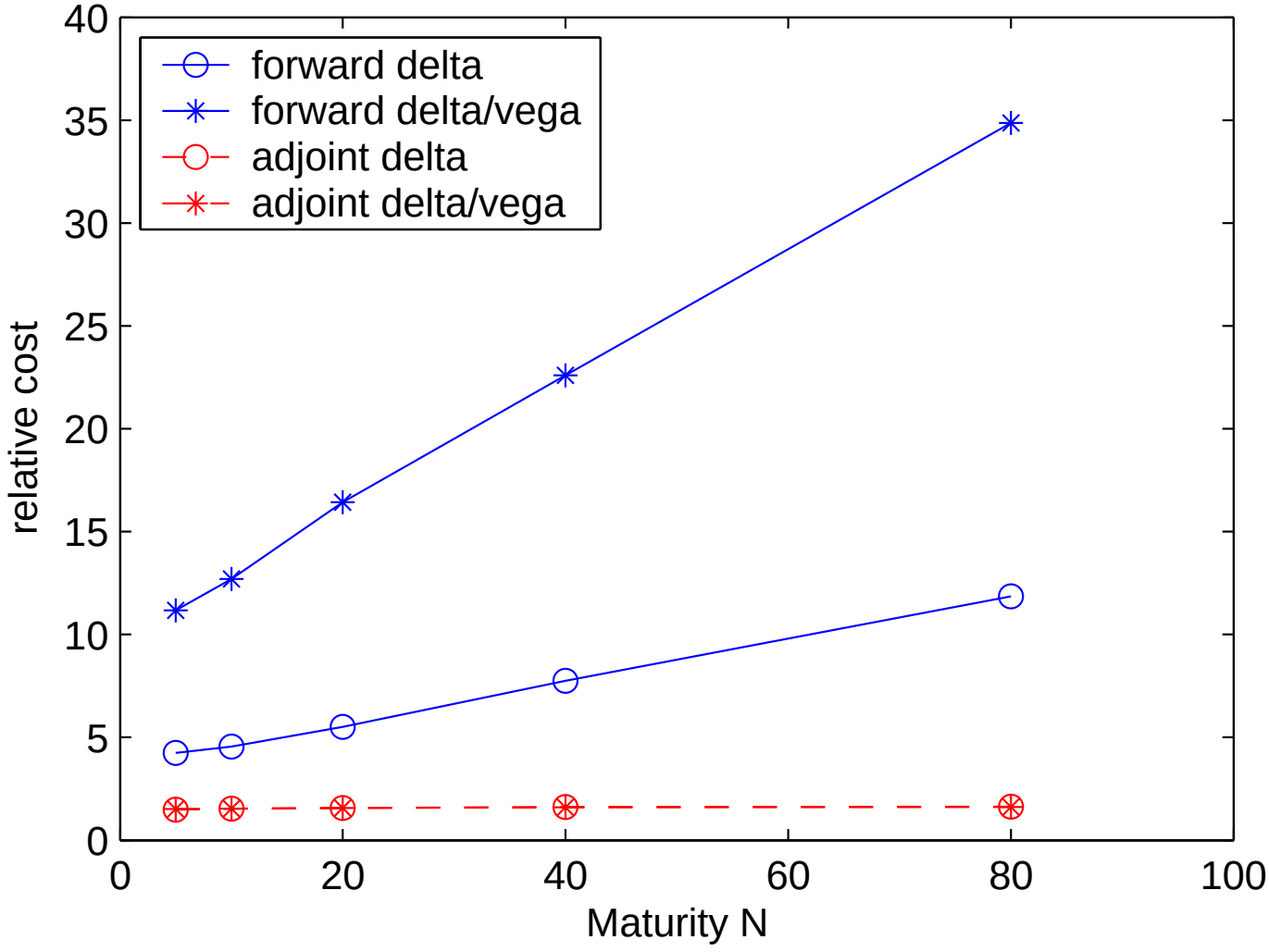
LMM results

Finite differences versus forward pathwise sensitivities:



LMM results

Forward versus adjoint pathwise sensitivities:



Generic adjoint approach

The adjoint (or dual or reverse mode) approach computes the same values as the forward pathwise approach, but much more efficiently for the sensitivity of a single output to multiple inputs.

The approach has a long history in applied math and engineering:

- optimal control theory (find control which achieves target and minimizes cost);
- design optimization (find shape which maximizes performance);
- primal/dual variables in linear programming optimization.

Generic adjoint approach

Returning to the generic stochastic o.d.e.

$$\mathrm{d}X = a(X)\,\mathrm{d}t + b(X)\,\mathrm{d}W,$$

an Euler approximation with timestep h gives

$$X(n{+}1) = F_n(X(n)) \equiv X(n) + a(X(n))\,h + b(X(n))\,Z(n{+}1)\,\sqrt{h}.$$

Defining $\Delta(n) = \dfrac{\partial X(n)}{\partial X(0)}$, then

$$\Delta(n{+}1) = D(n)\,\Delta(n), \qquad D(n) \equiv \frac{\partial F_n(X(n))}{\partial X(n)},$$

and hence

$$\frac{\partial g(X(N))}{\partial X(0)} = \frac{\partial g(X(N))}{\partial X(N)}\Delta(N) = \frac{\partial g}{\partial X} D(N{-}1)D(N{-}2)\ldots D(0)\Delta(0)$$

Generic adjoint approach

If X is m-dimensional, then $D(n)$ is an $m \times m$ matrix, and the overall computational cost is $O(Nm^3)$.

Alternatively,

$$\frac{\partial g(X(N))}{\partial X(0)} = \frac{\partial g}{\partial X}\, D(N{-}1)\, D(N{-}2) \cdots D(0)\, \Delta(0) = V(0)^\top \Delta(0),$$

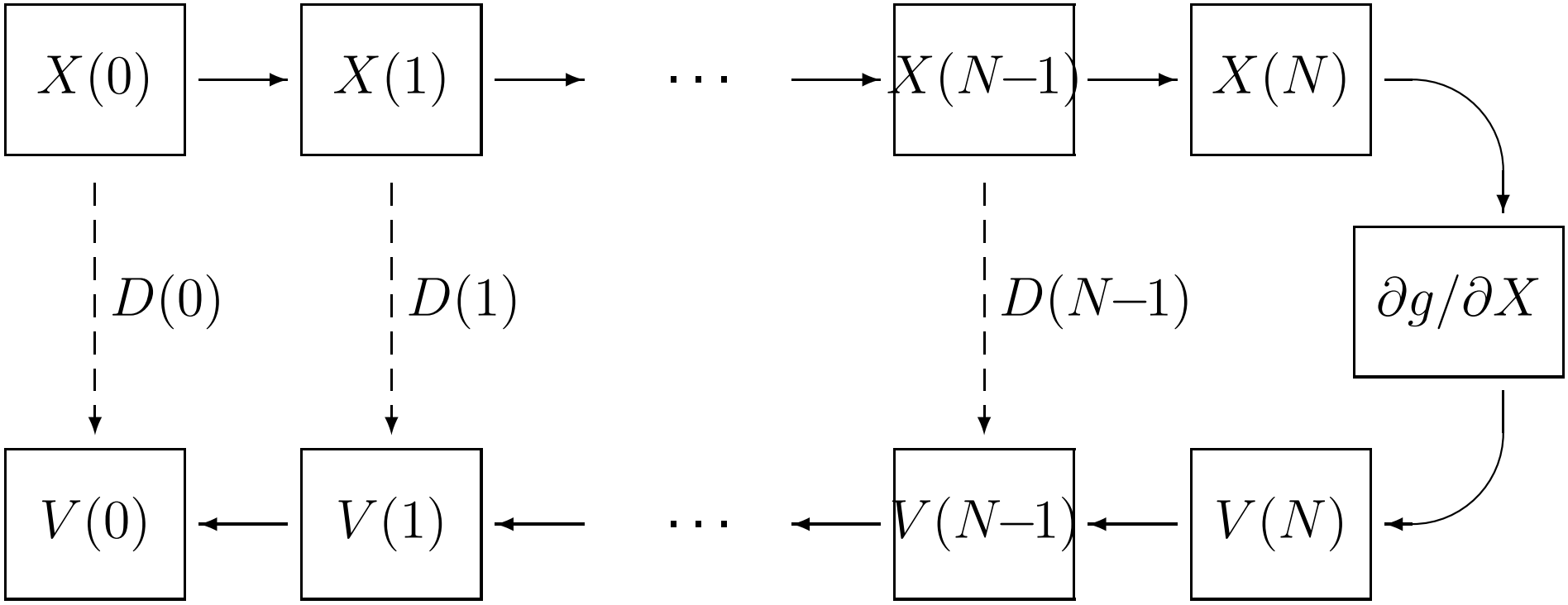
where adjoint $V(n) = \left(\dfrac{\partial g(X(N))}{\partial X(n)} \right)^\top$ is calculated from

$$V(n) = D(n)^\top V(n{+}1), \quad V(N) = \left(\frac{\partial g}{\partial X(N)} \right)^\top,$$

at a computational cost which is $O(Nm^2)$.

Generic adjoint approach

Note the flow of data within the path calculation:



– memory requirements are not significant because data only needs to be stored for the current path.

Generic adjoint approach

To calculate the sensitivity to other parameters, consider a generic parameter θ. Defining $\Theta(n) = \partial X(n)/\partial\theta$, then

$$\Theta(n+1) = \frac{\partial F_n}{\partial X}\,\Theta(n) + \frac{\partial F_n}{\partial \theta} \equiv D(n)\,\Theta(n) + B(n),$$

and hence

$$\begin{aligned}
\frac{\partial g}{\partial \theta} &= \frac{\partial g}{\partial X(N)}\,\Theta(N) \\
&= \frac{\partial g}{\partial X(N)}\Big\{B(N-1) + D(N-1)B(N-2) + \ldots \\
&\qquad\qquad + D(N-1)D(N-2)\ldots D(1)B(0)\Big\} \\
&= \sum_{n=0}^{N-1} V(n+1)^{\top} B(n).
\end{aligned}$$

Generic adjoint approach

Different θ's have different B's, but same V's

$$\Longrightarrow \text{ Computational cost} \simeq Nm^2 \; + \; Nm \times \#\text{ parameters},$$

compared to the standard forward approach for which

$$\text{Computational cost} \simeq Nm^2 \times \#\text{ parameters}.$$

However, the adjoint approach only gives the sensitivity of one output, whereas the forward approach can give the sensitivities of multiple outputs for little additional cost.

Generic adjoint approach

Defining $G(n) = \partial^2 X(n)/\partial X_j(0)\partial X_k(0)$ for a particular (j,k) it can be shown that

$$G(n+1) = D(n)\,G(n) + C(n),$$

where $C(n)$ is a complicated quadratic function of $\Delta(n)$.

Hence, pathwise Gammas can be computed efficiently by doing a forward calculation of Δ, followed by an adjoint calculation to compute

$$\sum_{n=0}^{N-1} V(n+1)^\top C(n),$$

for each pair (j,k), at a savings of factor $O(m)$ relative to a standard forward approach.

LMM implementation

The generic description shows the potential for significant savings, but real implementations can exploit features of specific applications for additional savings.

For the LIBOR market model, this gives a factor m savings:

Cost per timestep	Value	forward Deltas	adjoint Deltas
Generic	$O(m^2)$	$O(m^3)$	$O(m^2)$
Optimized	$O(m)$	$O(m^2)$	$O(m)$

LMM implementation

Applying the Euler scheme to the logarithms of the forward rates yields

$$L_i(n+1) = L_i(n) \exp\left([\mu_i(L(n)) - \|\sigma_i\|^2/2]h + \sigma_i^\top Z(n+1)\sqrt{h} \right).$$

For efficiency, we first compute

$$S_i(n) = \sum_{k=\eta(t)}^{i} \frac{\sigma_k \delta_k L_k(n)}{1 + \delta_k L_k(n)},$$

and then obtain $\mu_i = \sigma_i^\top S_i$.

Each timestep, there is an $O(m)$ cost in computing the S_i's, and then an $O(m)$ cost in updating the L_i's.

LMM implementation

Defining $\Delta_{ij}(n) = \partial L_i(n)/\partial L_j(0)$, differentiating the Euler scheme yields

$$\Delta_{ij}(n+1) = \frac{L_i(n+1)}{L_i(n)}\,\Delta_{ij}(n) + L_i(n+1)\,\sigma_i^\top\,S_{ij}(n)\,h,$$

where

$$S_{ij}(n) = \sum_{k=\eta(nh)}^{i} \frac{\sigma_k\,\delta_k\,\Delta_{kj}(n)}{(1+\delta_k L_k(n))^2}.$$

Each timestep, there is an $O(m^2)$ cost in computing the S_{ij}'s, and then an $O(m^2)$ cost in updating the Δ_{ij}'s.

(Note: programming implementation requires only multiplication and addition – very rapid on modern CPU's).

LMM implementation

Working through the details of the adjoint formulation, one eventually finds that $V_i(n) = V_i(n+1)$ for $i < \eta(nh)$, and

$$V_i(n) = \frac{L_i(n+1)}{L_i(n)} V_i(n+1) + \frac{\sigma_i^\top \delta_i \, h}{(1+\delta_i L_i(n))^2} \sum_{j=i}^{m} L_j(n+1) V_j(n+1) \sigma_j$$

for $i \geq \eta(nh)$.

Each timestep, there is an $O(m)$ cost in computing the summations, and then an $O(m)$ cost in updating the V_i's.

The correctness of the formulation is verified by checking it gives the same sensitivities as the forward calculation.

Algorithmic/automatic differentiation

- This LMM example is not “lucky”: an efficient evaluation $\Longrightarrow$ an efficient adjoint implementation;
- This is guaranteed by a theoretical result from the field of *algorithmic differentiation* which applies generic ideas to computer codes;
- Going further, *automatic differentiation* tools generate new computer code to perform standard (forward mode) and adjoint (reverse mode) sensitivity calculations;
- For more information see `http://www.autodiff.org`

Conclusions and future work

- Adjoint approach gives an unlimited number of sensitivities for a cost comparable to the initial valuation;
- The adjoint approach can also be used with PDE methods – it is applicable whenever one is interested in the sensitivity of one output to many inputs;
- AD tools may reduce the implementation effort;
- Payoff regularization may handle the problem of discontinuous payoffs;
- Issues of model calibration need to be addressed.